A
PROJECT
ON

HOME AUTOMATION USING ARDUINO WIFI MODULE ESP8266

Submitted in partial fulfillment of the requirements for the Diploma in electrical engineering

BY

BASANT LAL VISHWAKARMA - PEEG17118

UNDER THE GUIDANCE OF

Mrs BARNALI MOTLING

K.J Somaiya polytechnic, Vidyavihar Mumbai

ACADEMIC YEAR: 2021-2022

DECLARATION

We hereby declare that the project entitled ***"HOME AUTOMATION USING ARDUINO WIFI MODULE ESP8266"*** submitted for the Diploma in Electrical is our original work and the project has not formed the basis for the award of any associate ship, fellowship or any other similar titles.

Signature of the Student

ABSTRACT

This project presents a design and prototype implementation of new home automation system that uses Wi-Fi technology as a network infrastructure connecting its parts. The proposed system consists of two main components; the first part is the server (web server), which presents system core that manages, controls, and monitors users' home.

Users and system administrator can locally (LAN) or remotely (internet) manage and control system code. Second part is hardware interface module, which provides appropriate interface to sensors and actuator of home automation system.

Unlike most of available home automation system in the market the proposed system is scalable that one server can manage many hardware interface modules as long as it exists on Wi-Fi network coverage. System supports a wide range of home automation devices like power management components, and security components.

The proposed system is better from the scalability and flexibility point of view than the commercially available home automation systems.

TABLE OF CONTENTS

CONTENTS

1. INTRODUCTION

2. OBJECTIVE OF PROJECT

3. ESP8266 RELAY MODULE WEB SERVER USING ARDUINO IDE (8 Channels).

   3.1- *Short Introducing of ESP8266*
   3.2- *About the ESP8266*
   3.3- *ESP8266 Specifications*
   3.4- *ESP8266 Pinout*

4. INTRODUCING RELAY AND NETWORK CREDENTIALS.
   4.1- *Control relay module with 8 channels using an ESP8266 web server.*
   4.2- *Installing the ESp8266 asyncwebserver, Asynctcp and ESpasyncctcp libraries.*
   4.3- *Define relay configuration.*
   4.4- *Define pin assignment.*
   4.5- *Network Credentials.*
   4.6- *Wiring a relay module to an ESP8266 board.*

5. DEMONSTRATION.

6. ENCLOSURE FOR SEFTY.

7. CONCLUSION

8. REFERENCES

1. INTRODUCTION

A load controlled by computer systems has many advantages compared with manual controlled loads. Nowadays there are many programs and applications help to control things better using codes or python algorithms in artificial intelligence projects. In order to save energy and make loads monitored easily, this research suggests smart home project based on IoT technology. This smart home is an Internet of Things (IoT) project that controls loads with internet connection via Wireless Fidelity WIFI connection. A smart phone connected to internet with Arduino IDE application as a control panel, and NodeMCU microcontroller kit in other side as a controller that receives control commands via WIFI signal. NodeMCU kit is built with ESP8266 WIFI receiver that able to process and analyze WIFI signal to input the microcontroller. The WIFI receiver and microcontroller are built in one kit to be used as IoT project. It's called NodeMCU.

To connect the system to the Internet, needs a Wi-Fi receiver. In my case I used ESP8266 that is connected as built-in in the NodeMCU board that contains a firmware runs with the ESP8266. The firmware is a low-level control computer software.

The NodeMCU is coded via Arduino Integrated Development Environment (IDE) with the Universal Serial Bus port (USB) to tell the NodeMCU what to do, I want to make the NodeMCU controls four-channel relay kit by Arduino hand phone application and shows the temperature that measured by LM35 sensor.

Parts used to create the project:

1) NodeMCU board. Open source internet of things platform.

2) 5V DC supply for relay and board.

3) 8-channel relay kit. To drive loads from digital NodeMCU output pins.

6) Computer with Arduino (IDE) program installed to code the NodeMCU once.

2. OBJECTIVE OF PROJECT

The goal of this project is to develop a home automation system that gives the user complete control over all remotely controllable aspects of his or her home.

The automation system will have the ability to be controlled from a central host PC, the Internet, and also remotely accessed via a Pocket PC with a Windows Mobile based application.

The System will also sense the Accidental Gas leakage, water level and will notify the user by SMS.

3. ESP8266 Relay Module Web Server using Arduino IDE (8 Channels)

This tutorial is a step-by-step guide that covers how to build a standalone ESP32 or ESP8266 NodeMCU Web Server that controls any relay module. We'll create an ESP8266 Web Server that is mobile responsive and it can be accessed with any device with a browser in your local network.

3.1. *Short Introducing of ESP8266*

The ESP8266 is a $4 (up to $10) Wi-Fi module. It allows you to control inputs and outputs as you would do with an Arduino, but it comes with Wi-Fi.
So, it is great for home automation/internet of things applications.

- create a web server
- send HTTP requests
- control outputs
- read inputs and interrupts

3.2. *About the ESP8266*

- 11 b/g/n protocol
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Built-in low-power 32-bit CPU
- SDIO 2.0, SPI, UART

3.3. *ESP8266 Specifications*

The most widely used ESP8266 development boards are the ESP-01, the ESP8266-12E NodeMCU Kit and the Wemos D1 Mini. We'll show you the pinout for those boards. If you're using another development board, make sure you have the right pinout.

3.4. ESP8266 Pinout.

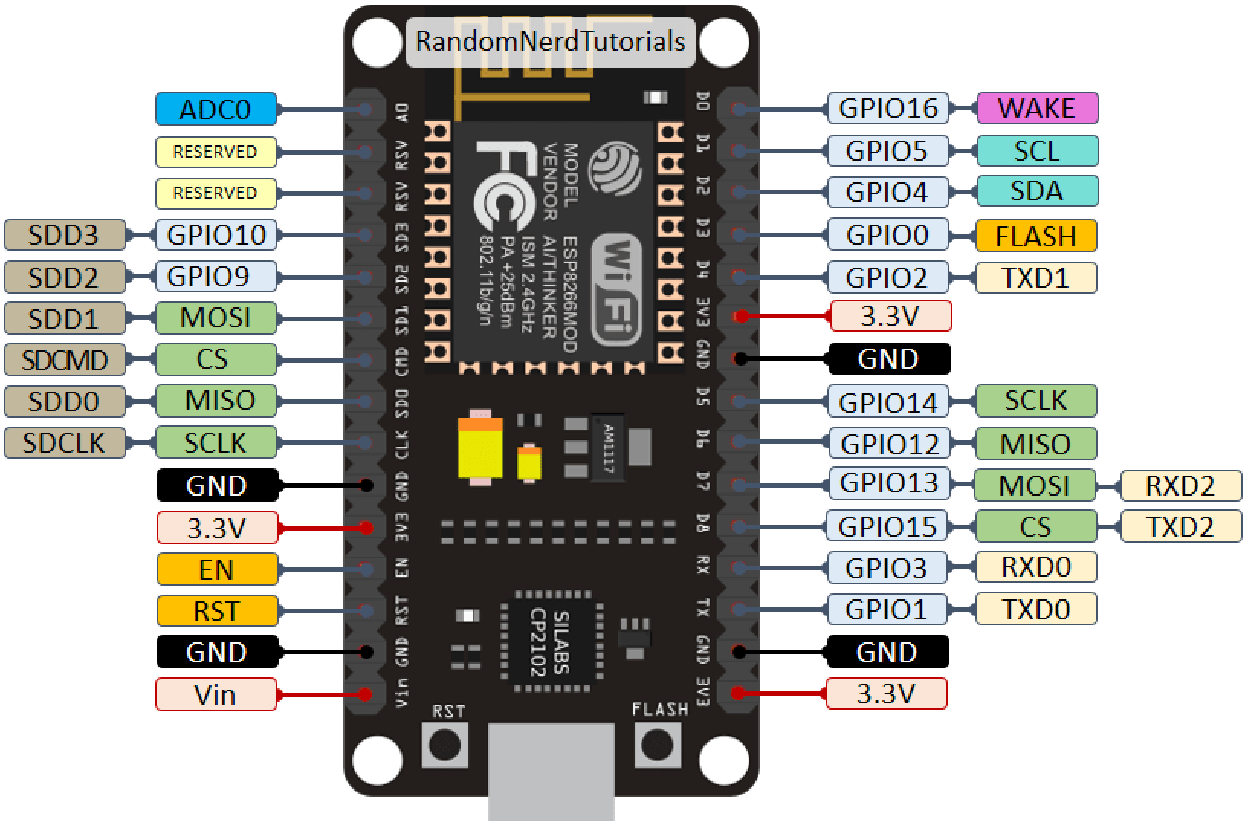


Fig. 1

4. *Introducing Relays and Network credentials.*

A relay is an electrically operated switch and like any other switch, it that can be turned on or off, letting the current go through or not. It can be controlled with low voltages, like the 3.3V provided by the ESP GPIOs and allows us to control high voltages like 12V, 24V or mains voltage (230V in Europe and 120V in the US).

There are different relay modules with a different number of channels. You can find relay modules with one, two, four, eight and even sixteen channels. The number of channels determines the number of outputs you can control.

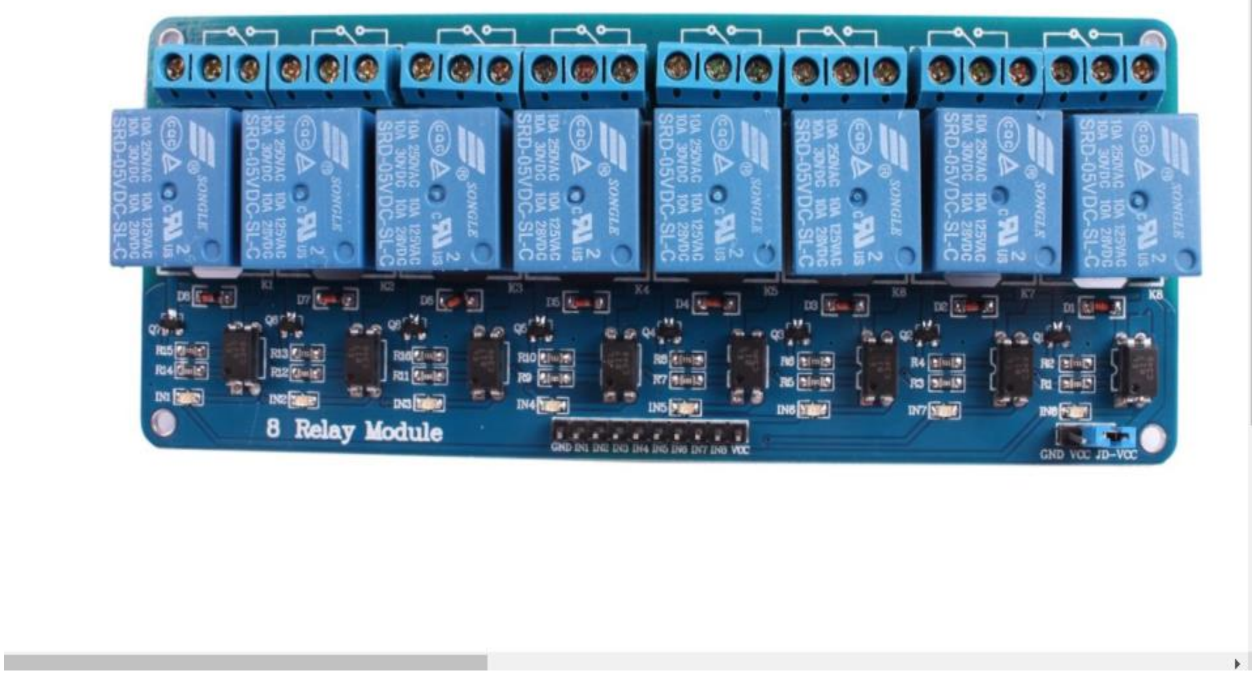


Fig. 2

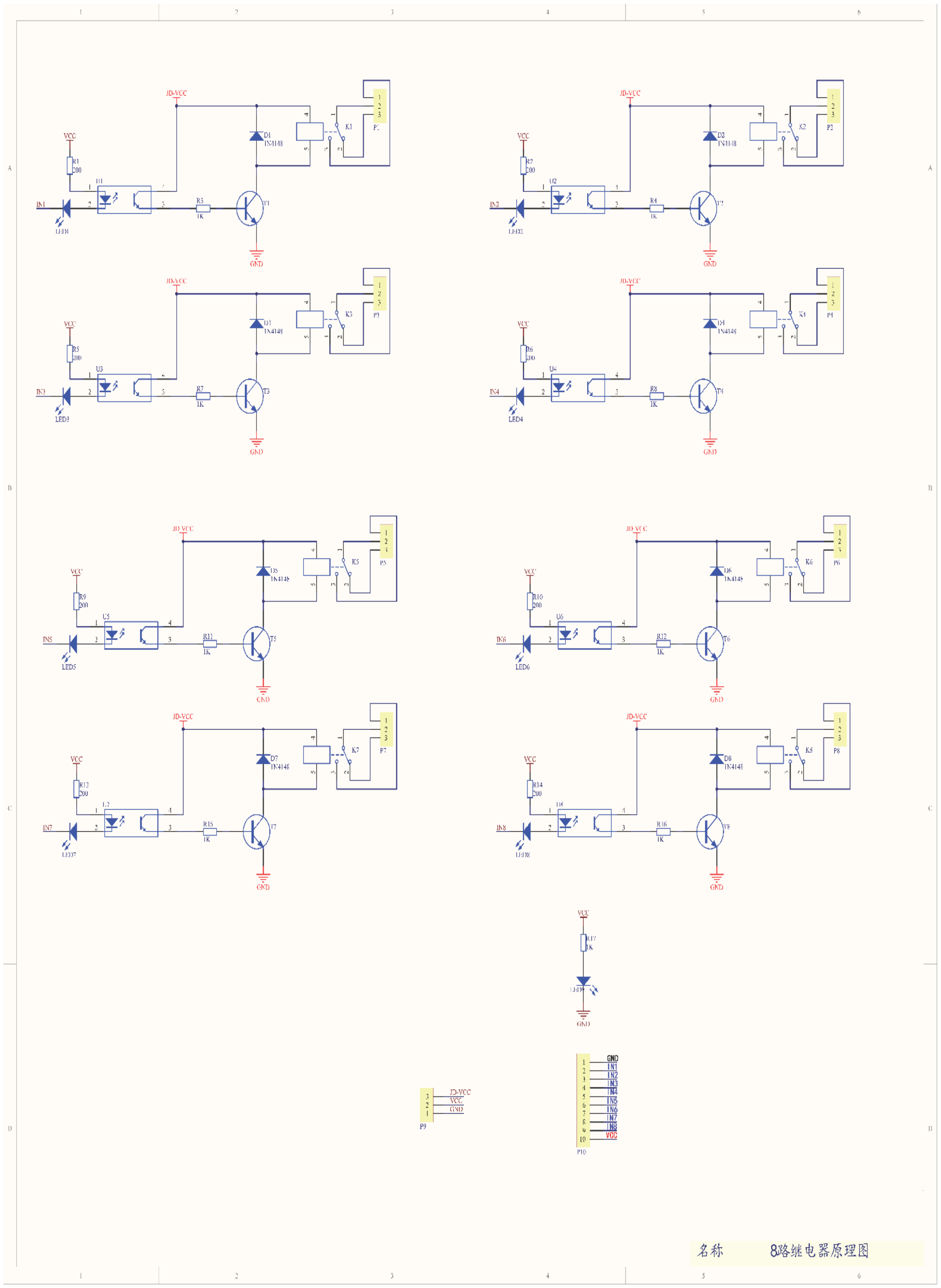
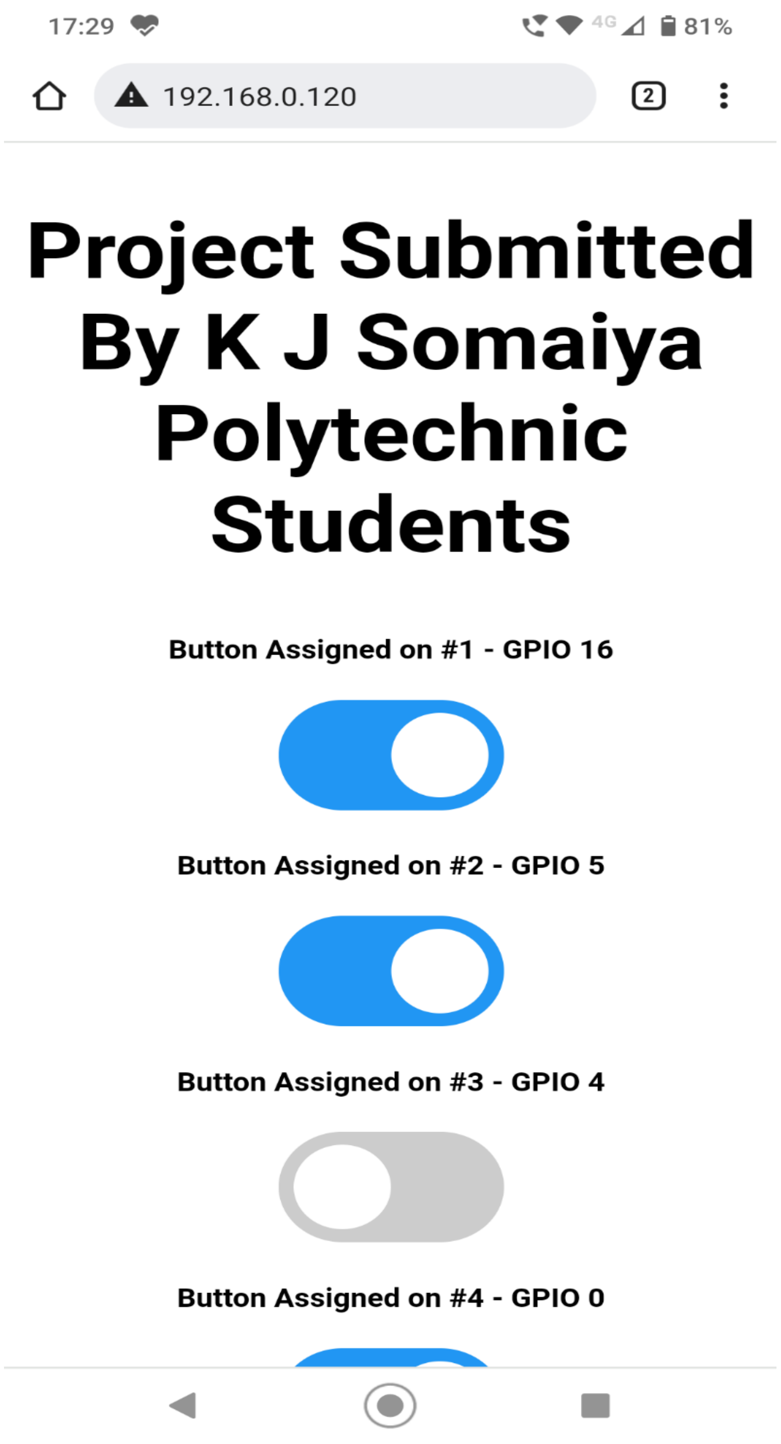

JD-VCC
VCC
GND
IN1
IN2
IN3
IN4
IN5
IN6
IN7
IN8
K1
K2
K3
K4
K5
K6
K7
K8
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
1N4148
1K
200
名称
8路继电器原理图

Fig.3

4.1. *Control Relay Modules with 8 Channels using an ESP8266 Web Server*



17:29 81%

192.168.0.120

Project Submitted By K J Somaiya Polytechnic Students

Button Assigned on #1 - GPIO 16

Button Assigned on #2 - GPIO 5

Button Assigned on #3 - GPIO 4

Button Assigned on #4 - GPIO 0

Fig. 4

With this web server code, you can control as many relays as you want via web server whether they are configured as normally opened or as normally closed. You just need to change a few lines of code to define the number of relays you want to control and the pin assignment.

To build this web server, install the ESP8266 board add-on and the next libraries in your Arduino IDE.

4.2. Installing the ESPAsyncWebServer, AsyncTCP, and ESPAsyncTCP Libraries

In your Arduino IDE, to install the libraries go to **Sketch** > **Include Library** > **Add .ZIP library…** and select the library you've just downloaded.

After installing the required libraries, copy the following code to your Arduino IDE

```
/*********

                                   A

                                Project

                               Submitted

              Home Automation based on ESP Web server (ESP8266)

 #Submitted in partial fulfillment of the requirements for the Diploma in electrical engineering


                                  BY


              BASANT LAL VISHWAKARMA:          PEEG17118
```

```
          UNDER THE GUIDANCE OF

           Mrs BARNALI MOTLING
From K J Somaiya Polytechic Vidyavihar Mumbai- 400077
        ACADEMIC YEAR:  2021-2022
 The above copyright notice and this permission notice shall be included in all
  copies or substantial portions of the Software.
*********/
// Import required libraries
#include "ESP8266WiFi.h"
#include "ESPAsyncWebServer.h"

// Set to true to define Relay as Normally Open (NO)
#define RELAY_NO    true
// Set number of relays
#define NUM_RELAYS  8
// Assign each GPIO to a relay
int relayGPIOs[NUM_RELAYS] = {16, 05, 04, 00, 02, 14, 12, 13};
// Replace with your network credentials
const char* ssid = "__Guest_User__";
const char* password = "Panasonic@98765";

const char* PARAM_INPUT_1 = "relay";
const char* PARAM_INPUT_2 = "state";
// Create AsyncWebServer object on port 80
```

```
AsyncWebServer server(80);

const char index_html[] PROGMEM = R"rawliteral(

<!DOCTYPE HTML><html>

<head>

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <style>

    html {font-family: Arial; display: inline-block; text-align: center;}

    h2 {font-size: 3.0rem;}

    p {font-size: 3.0rem;}

    body {max-width: 600px; margin:0px auto; padding-bottom: 25px;}

    .switch {position: relative; display: inline-block; width: 120px; height: 68px}

    .switch input {display: none}

    .slider {position: absolute; top: 0; left: 0; right: 0; bottom: 0; background-color: #ccc; border-radius: 
34px}

    .slider:before {position: absolute; content: ""; height: 52px; width: 52px; left: 8px; bottom: 8px; 
background-color: #fff; -webkit-transition: .4s; transition: .4s; border-radius: 68px}

    input:checked+.slider {background-color: #2196F3}

    input:checked+.slider:before {-webkit-transform: translateX(52px); -ms-transform: translateX(52px); 
transform: translateX(52px)}

  </style>

</head>

<body>

  <h2>Project Submitted By K J Somaiya Polytechnic Students</h2>

  %BUTTONPLACEHOLDER%

<script>function toggleCheckbox(element) {

  var xhr = new XMLHttpRequest();

  if(element.checked){ xhr.open("GET", "/update?relay="+element.id+"&state=1", true); }
  else { xhr.open("GET", "/update?relay="+element.id+"&state=0", true); }
  xhr.send();
}</script>
</body>
</html>
)rawliteral";

// Replaces placeholder with button section in your web page
String processor(const String& var){
  //Serial.println(var);
  if(var == "BUTTONPLACEHOLDER"){
    String buttons ="";
    for(int i=1; i<=NUM_RELAYS; i++){
      String relayStateValue = relayState(i);
      buttons+= "<h4>Button Assigned on #" + String(i) + " - GPIO " + relayGPIOs[i-1] + "</h4><label
class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"" + String(i) + "\"
"+ relayStateValue +"><span class=\"slider\"></span></label>";
    }
    return buttons;
  }
  return String();
}
String relayState(int numRelay){
  if(RELAY_NO){
    if(digitalRead(relayGPIOs[numRelay-1])){
      return "";
    }
    else {
      return "checked";
    }
  }
  else {
    if(digitalRead(relayGPIOs[numRelay-1])){
      return "checked";
    }
    else {
      return "";
    }
  }
  return "";
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);

  // Set all relays to off when the program starts - if set to Normally Open (NO), the relay is off when you
set the relay to HIGH
  for(int i=1; i<=NUM_RELAYS; i++){
    pinMode(relayGPIOs[i-1], OUTPUT);
    if(RELAY_NO){
      digitalWrite(relayGPIOs[i-1], HIGH);
    }
    else{
      digitalWrite(relayGPIOs[i-1], LOW);
    }
  }
    // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }
  // Print ESP8266 Local IP Address
  Serial.println(WiFi.localIP());

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });

  // Send a GET request to <ESP_IP>/update?relay=<inputMessage>&state=<inputMessage2>
  server.on("/update", HTTP_GET, [] (AsyncWebServerRequest *request) {
    String inputMessage;
    String inputParam;
    String inputMessage2;
    String inputParam2;
```

```
    // GET input1 value on <ESP_IP>/update?relay=<inputMessage>
    if (request->hasParam(PARAM_INPUT_1) & request->hasParam(PARAM_INPUT_2)) {
      inputMessage = request->getParam(PARAM_INPUT_1)->value();
      inputParam = PARAM_INPUT_1;
      inputMessage2 = request->getParam(PARAM_INPUT_2)->value();
      inputParam2 = PARAM_INPUT_2;
      if(RELAY_NO){
        Serial.print("NO ");
        digitalWrite(relayGPIOs[inputMessage.toInt()-1], !inputMessage2.toInt());
      }
      else{
        Serial.print("NC ");
        digitalWrite(relayGPIOs[inputMessage.toInt()-1], inputMessage2.toInt());
      }
    }
    else {
      inputMessage = "No message sent";
      inputParam = "none";
    }
    Serial.println(inputMessage + inputMessage2);
    request->send(200, "text/plain", "OK");
  });
  // Start server
  server.begin();
}
```

```
void loop() {

}
```

4.3. Define Relay Configuration

Modify the following variable to indicate whether you're using your relays in normally open (NO) or normally closed (NC) configuration. Set the RELAY_NO variable to true for normally open os set to false for normally closed.

```
#define RELAY_NO true
```

4.4. Define Relays Pin Assignment

In the following array variable you can define the ESP GPIOs that will control the relays.

```
int relayGPIOs[NUM_RELAYS] = {16, 05, 04, 00, 02, 14, 12, 13};
```

The number of relays set on the NUM_RELAYS variable needs to match the number of GPIOs assigned in the relayGPIOs array.

4.5. Network Credentials

Insert your network credentials in the following variables.

```
const char* ssid     = "Your WIFI SSID (local)";
const char* password = "Your WIFI PASSWORD";
```

4.6. *Wiring a Relay Module to an ESP8266 Board*

For demonstration purposes, we're controlling 8 relay channels. Wire the ESP8266 boards to the relay module as shown in the next schematic diagrams.

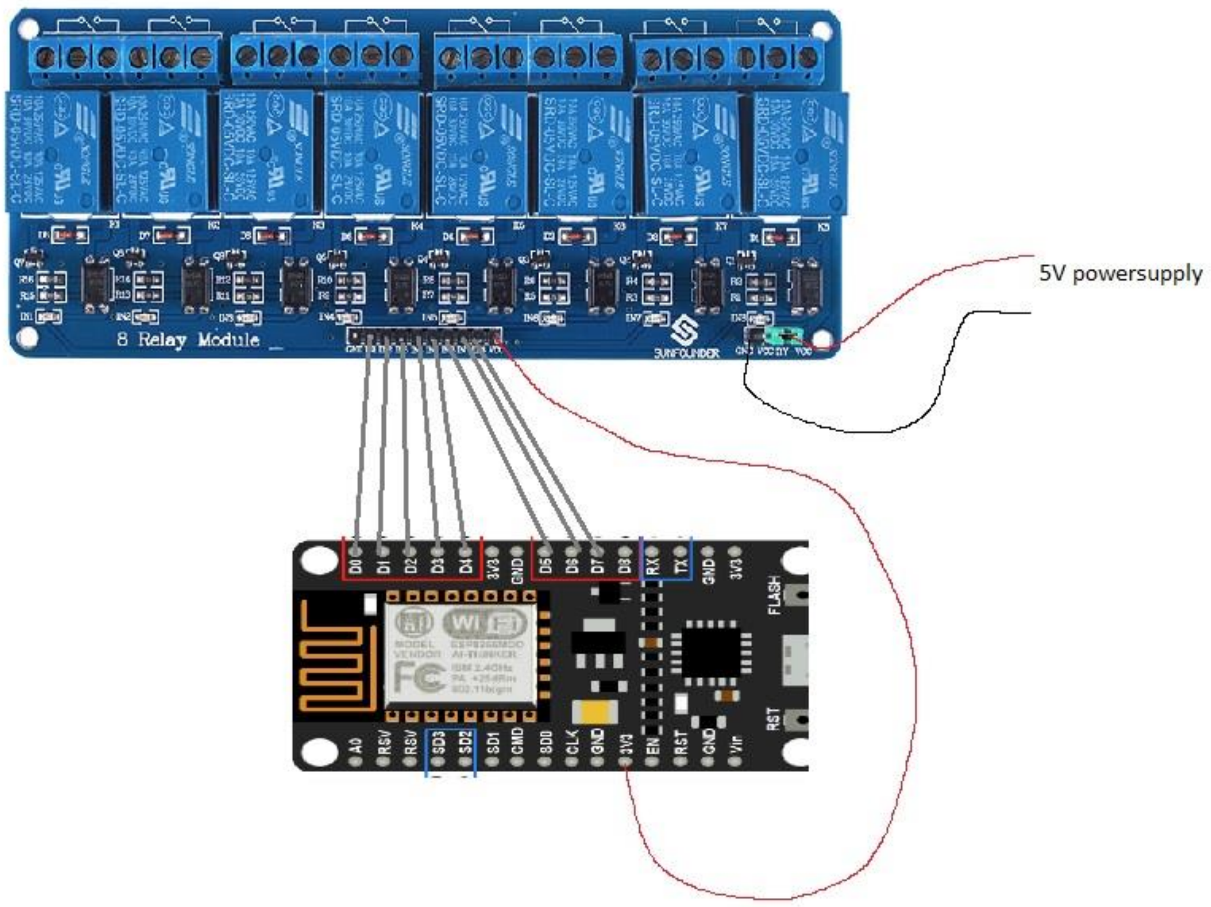


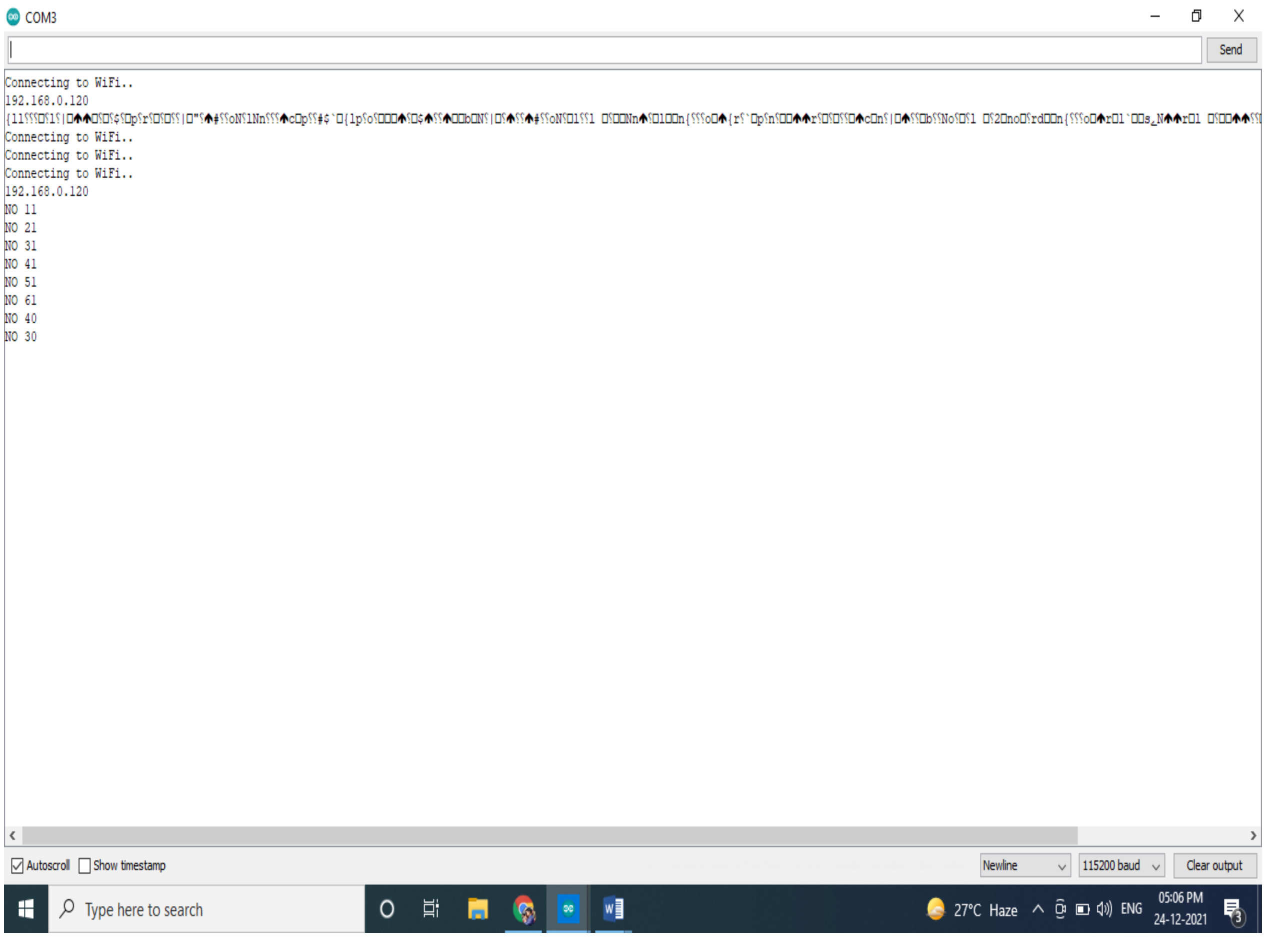
Fig. 5

5. Demonstration

After making the necessary changes, upload the code to your ESP. Open the Serial Monitor at a baud rate of 115200 and press the ESP8266 RST button to get its IP address (*make sure that you circuit is unplugged from mains voltage*).

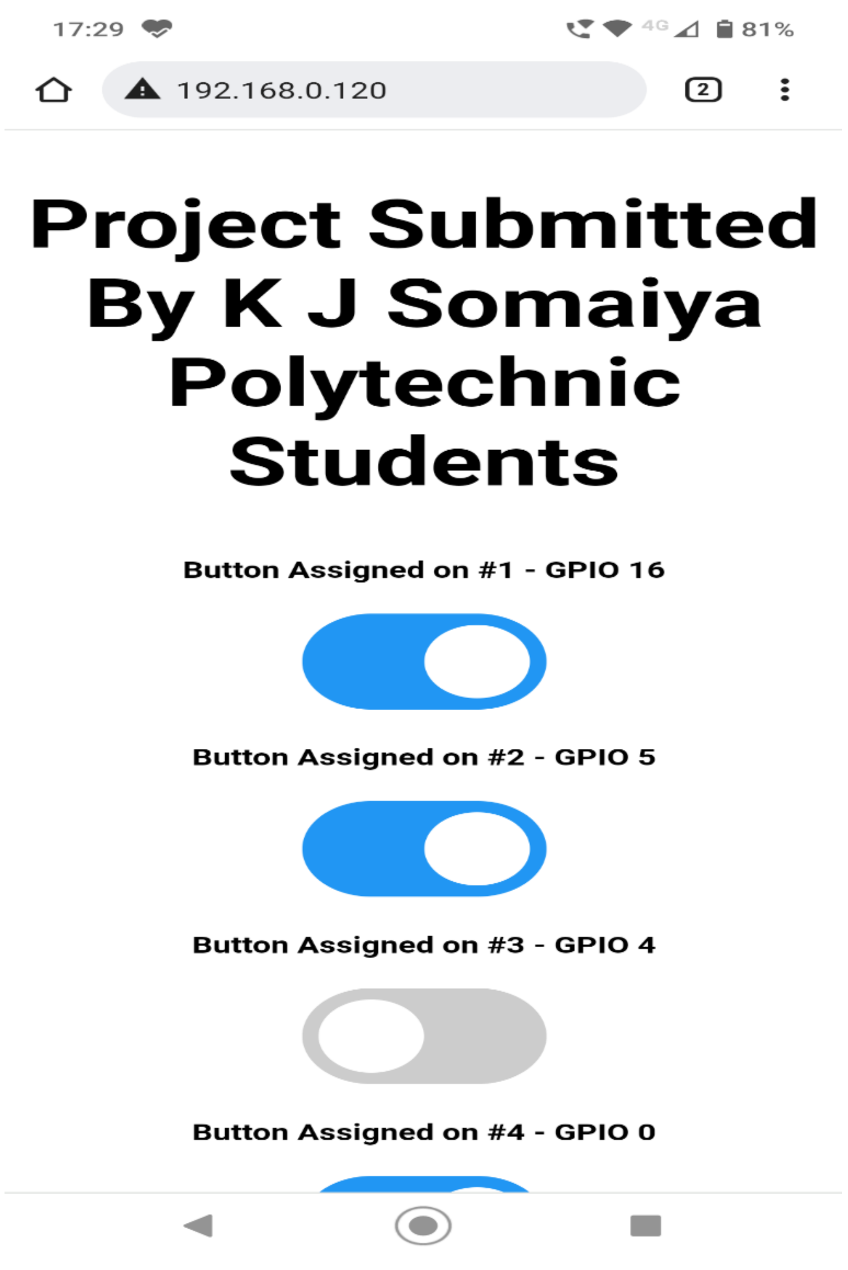
Open a browser in your local network and type the ESP8266 IP address to get access to the web server. You should get something as follows with as many buttons as the number of relays you've defined in your code.

Now, you can use the buttons to control your relays remotely using your smartphone.



```
Connecting to WiFi..
192.168.0.120
[illegible]
Connecting to WiFi..
Connecting to WiFi..
Connecting to WiFi..
192.168.0.120
NO 11
NO 21
NO 31
NO 41
NO 51
NO 61
NO 40
NO 30
```

Fig. 6


17:29 81%

192.168.0.120

Project Submitted By K J Somaiya Polytechnic Students

Button Assigned on #1 - GPIO 16

Button Assigned on #2 - GPIO 5

Button Assigned on #3 - GPIO 4

Button Assigned on #4 - GPIO 0

Fig. 7

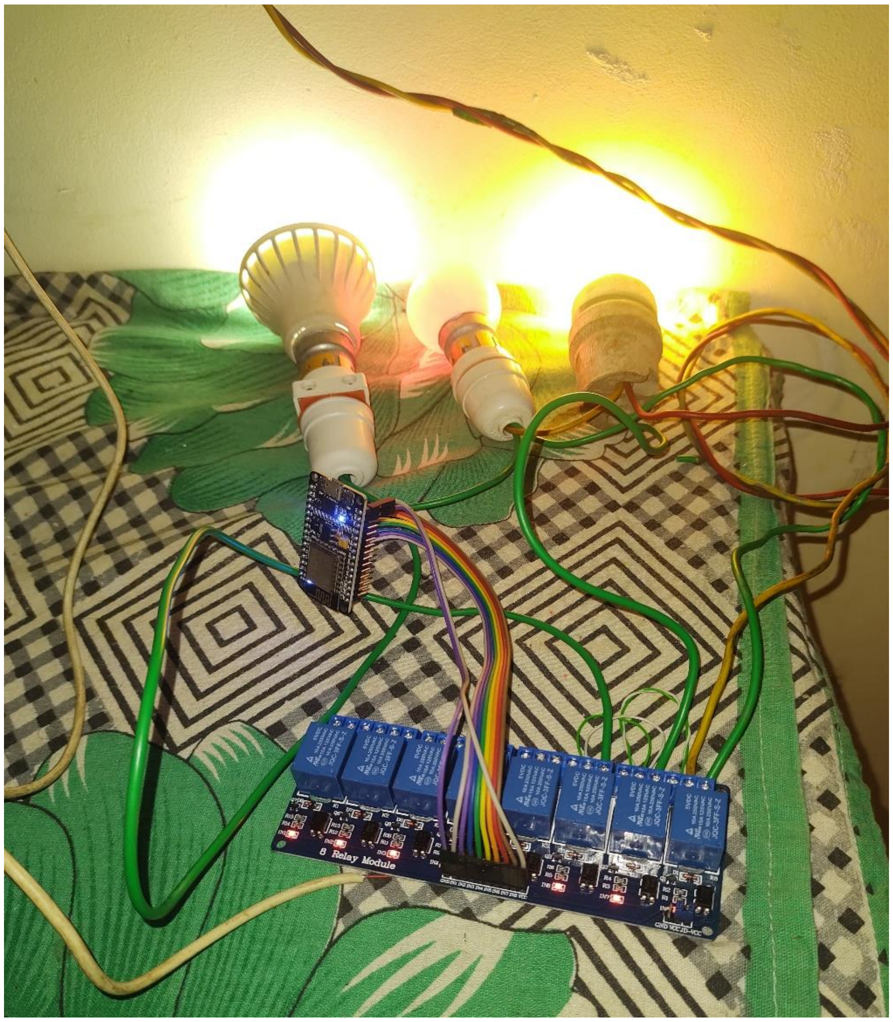


Fig. 8

Note: Only 3 lamps are available, but all relay and pin configure

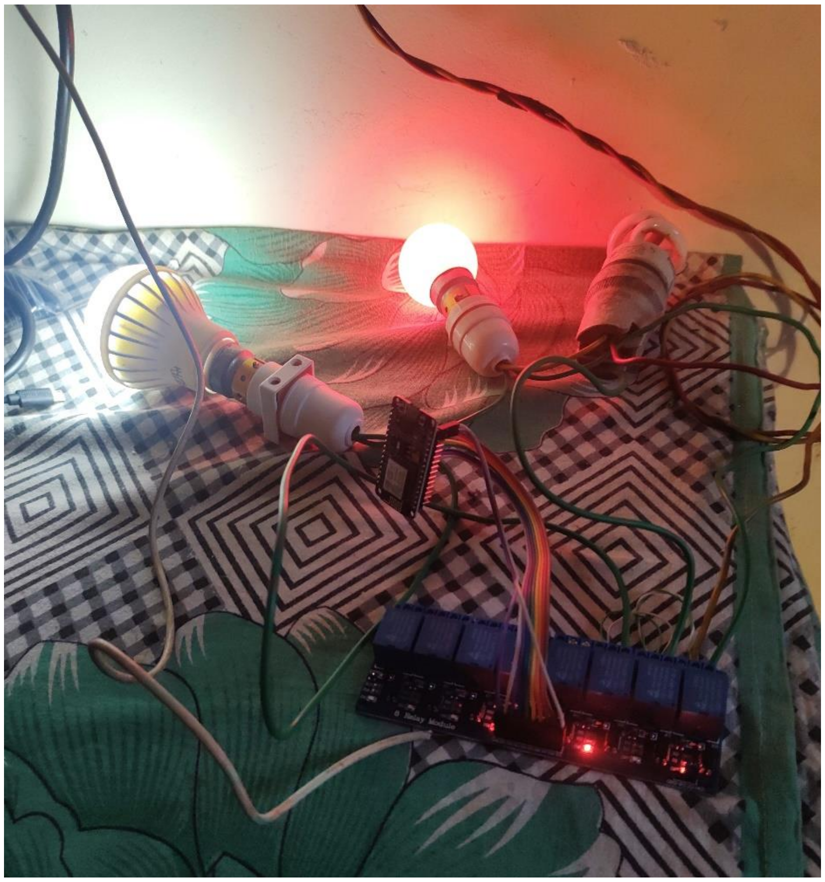


Fig. 9

6. Enclosure for Safety

For a final project, make sure you place your relay module and ESP inside an enclosure to avoid any AC pins exposed.

7. Conclusion

Based on the results of analysis of all data obtained by testing the smart home with the Internet of Things based NodeMCU ESP6288 module, the following conclusions can be drawn:
1) Smart Home with Internet of Things (IoT) based NodeMCU ESP8266 Module can be designed with various components hardware and software support so that it can be arranged into a smart home system that is controlled with the ARDUINO with chrome on android application according to what is intended.

2) The Smart Home with this Internet of Things (IoT) based NodeMCU ESP8266 Module can be implemented to control some of the home electronics performance including lighting controls, fan control, temperature monitoring, early warning systems and etc.

8. References

1) Get full project on ESP8266
2) Arduino Temperature Sensor Using LM35. Groups of Electronics Hobbyist, Roboticist. We Developed Electronics Project Tutorials Make Open for Everyone.

3) BOHORA, Bharat; MAHARJAN, Sunil; SHRESTHA, Bibek Raj. IoT Based Smart Home Using Blynk Framework. Zerone Scholar, [S.l.], v. 1, n. 1, p. 26-30, Dec. 2016. ISSN 2542-2774. Google scholar.

4) DC-DC Step Down Converter Power Supply Provides Regulated 5VDC Output with Range Input of 10-32VDC, Model GTD21088L-1505-T2.

5) Home Automation Using Internet of Thing 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) Published: 2016. Google Scholar.

6) Internet of Things in Home Automation and Energy Efficient Smart Home Technologies Simon G. M. Koo Department of Computer Engineering, Santa Clara University, CA 95053, USA

7) Low Cost Implementation of Smart Home Automation Ravi Kishore Kodali Department of Electronics and Communication Engineering National Institute of Technology, Warangal, 506004 India

8) Mobile based home automation using Internet of Things (IoT) 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT) Published: 2015

9) NodeMCU Features and Pinout. A Brief Tutorial on the Introduction to NodeMCU V3.

10) Yoyosteven in Circuits Microcontrollers. NODEMCU 1.0 (ESP8266) CONTROLLED RELAY USING BLYNK (OVER THE WEB).

11) 5V 4-Channel Relay Interface Board, Standard Interface that can be Controlled Directly by Microcontroller.

12) 15-17 March 2018 U. Venkanna IoT Based Smart Home Automation System Using Sensor Node. Google Scholar.